PROJECT OVERVIEW

In this project, I will focus on data cleaning, imputation, analysis, and visualization to derive valuable insights for a business stakeholder.

Business understanding

Our company is expanding in to new industries to diversify its portfolio. Specifically, they are interested in purchasing and operating airplanes for commercial and private enterprises, In order to pursue this venture the company has to be in the know of potential risks of this venture. This includes the number of accidents that each aircraft has had over the years,the market demand and the safety of the aircraft. I have been charged with determining which aircraft are the lowest risk for the company to start this new business endeavor. I will thereby translate my findings into actionable insights that the head of the new aviation division can use to help decide which aircraft to purchase.

The data

In the data folder is a dataset from the National Transportation Safety Board that includes aviation accident data from 1962 to 2023 about civil aviation accidents and selected incidents in the United States and international waters.

1. My analysis would yield three concrete business recommendations
2. Communication
3. Visualization

Import necessary libraries to be used in the project

```
#IMPORTING ALL NECESSARY LIBRARIES.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
%matplotlib inline

#This part here ignores warnings
from pandas.errors import DtypeWarning

warnings.filterwarnings("ignore", category=DtypeWarning)
```

Load the dataset

```
#This reads the dataset that I want to work on
df = pd.read_csv("AviationData.csv",encoding = 'Windows-1252')
```

Data sanity check

```
#This gives us the shape of the dataset.It has (88889)rows and
(31)collumns
df.shape
```

```
(88889, 31)
```

```
#This give an outline of the first 5 rows of the dataset
df.head()
```

```
         Event.Id Investigation.Type Accident.Number  Event.Date  \
0  20001218X45444           Accident      SEA87LA080  1948-10-24
1  20001218X45447           Accident      LAX94LA336  1962-07-19
2  20061025X01555           Accident      NYC07LA005  1974-08-30
3  20001218X45448           Accident      LAX96LA321  1977-06-19
4  20041105X01764           Accident      CHI79FA064  1979-08-02

          Location        Country Latitude Longitude Airport.Code  \
0  MOOSE CREEK, ID  United States      NaN       NaN          NaN
1   BRIDGEPORT, CA  United States      NaN       NaN          NaN
2    Saltville, VA  United States  36.9222  -81.8781          NaN
3       EUREKA, CA  United States      NaN       NaN          NaN
4       Canton, OH  United States      NaN       NaN          NaN

  Airport.Name  ... Purpose.of.flight Air.carrier Total.Fatal.Injuries
\
0          NaN  ...          Personal         NaN                  2.0

1          NaN  ...          Personal         NaN                  4.0

2          NaN  ...          Personal         NaN                  3.0

3          NaN  ...          Personal         NaN                  2.0

4          NaN  ...          Personal         NaN                  1.0


  Total.Serious.Injuries Total.Minor.Injuries Total.Uninjured  \
0                    0.0                  0.0             0.0
1                    0.0                  0.0             0.0
2                    NaN                  NaN             NaN
3                    0.0                  0.0             0.0
4                    2.0                  NaN             0.0
```

```
  Weather.Condition  Broad.phase.of.flight   Report.Status
Publication.Date
0               UNK                 Cruise  Probable Cause
NaN
1               UNK                Unknown  Probable Cause       19-
09-1996
2               IMC                 Cruise  Probable Cause       26-
02-2007
3               IMC                 Cruise  Probable Cause       12-
09-2000
4               VMC               Approach  Probable Cause       16-
04-1980

[5 rows x 31 columns]
```

```
#This give an outline of the last 5 rows of the dataset
df.tail()
```

```
             Event.Id Investigation.Type Accident.Number
Event.Date  \
88884  20221227106491           Accident      ERA23LA093  2022-12-26

88885  20221227106494           Accident      ERA23LA095  2022-12-26

88886  20221227106497           Accident      WPR23LA075  2022-12-26

88887  20221227106498           Accident      WPR23LA076  2022-12-26

88888  20221230106513           Accident      ERA23LA097  2022-12-29


            Location        Country Latitude Longitude Airport.Code  \
88884  Annapolis, MD  United States      NaN       NaN          NaN
88885    Hampton, NH  United States      NaN       NaN          NaN
88886     Payson, AZ  United States  341525N  1112021W          PAN
88887     Morgan, UT  United States      NaN       NaN          NaN
88888     Athens, GA  United States      NaN       NaN          NaN

      Airport.Name  ... Purpose.of.flight        Air.carrier  \
88884          NaN  ...          Personal                NaN
88885          NaN  ...               NaN                NaN
88886       PAYSON  ...          Personal                NaN
88887          NaN  ...          Personal  MC CESSNA 210N LLC
88888          NaN  ...          Personal                NaN

      Total.Fatal.Injuries Total.Serious.Injuries Total.Minor.Injuries
\
88884                  0.0                    1.0                  0.0

88885                  0.0                    0.0                  0.0
```

```
88886                  0.0                     0.0                     0.0

88887                  0.0                     0.0                     0.0

88888                  0.0                     1.0                     0.0


      Total.Uninjured Weather.Condition  Broad.phase.of.flight
Report.Status  \
88884             0.0               NaN                   NaN
NaN
88885             0.0               NaN                   NaN
NaN
88886             1.0               VMC                   NaN
NaN
88887             0.0               NaN                   NaN
NaN
88888             1.0               NaN                   NaN
NaN

      Publication.Date
88884       29-12-2022
88885              NaN
88886       27-12-2022
88887              NaN
88888       30-12-2022

[5 rows x 31 columns]
```

```python
#Gives a look into the columns inside the dataframe
df.columns
```

```
Index(['Event.Id', 'Investigation.Type', 'Accident.Number',
'Event.Date',
       'Location', 'Country', 'Latitude', 'Longitude', 'Airport.Code',
       'Airport.Name', 'Injury.Severity', 'Aircraft.damage',
       'Aircraft.Category', 'Registration.Number', 'Make', 'Model',
       'Amateur.Built', 'Number.of.Engines', 'Engine.Type',
'FAR.Description',
       'Schedule', 'Purpose.of.flight', 'Air.carrier',
'Total.Fatal.Injuries',
       'Total.Serious.Injuries', 'Total.Minor.Injuries',
'Total.Uninjured',
       'Weather.Condition', 'Broad.phase.of.flight', 'Report.Status',
       'Publication.Date'],
      dtype='object')
```

```python
#This gives us the summary statistics of the dataset
df.describe()
```

```
       Number.of.Engines  Total.Fatal.Injuries  Total.Serious.Injuries
\
count       82805.000000          77488.000000            76379.000000

mean            1.146585              0.647855                0.279881

std             0.446510              5.485960                1.544084

min             0.000000              0.000000                0.000000

25%             1.000000              0.000000                0.000000

50%             1.000000              0.000000                0.000000

75%             1.000000              0.000000                0.000000

max             8.000000            349.000000              161.000000


       Total.Minor.Injuries  Total.Uninjured
count          76956.000000     82977.000000
mean               0.357061         5.325440
std                2.235625        27.913634
min                0.000000         0.000000
25%                0.000000         0.000000
50%                0.000000         1.000000
75%                0.000000         2.000000
max              380.000000       699.000000
```

```python
#This give summary satistics of objects
df.describe(include="object")
```

```
              Event.Id Investigation.Type Accident.Number  Event.Date
\
count            88889              88889           88889       88889

unique           87951                  2           88863       14782

top     20001214X45071           Accident       ERA22LA379  1982-05-16

freq                 3              85015               2          25


             Location        Country Latitude Longitude
Airport.Code  \
count           88837          88663    34382     34373        50249

unique          27758            219    25592     27156        10375

top     ANCHORAGE, AK  United States  332739N  0112457W         NONE
```

```
freq                434           82248       19         24          1488


       Airport.Name  ... Amateur.Built    Engine.Type FAR.Description
\
count         52790  ...         88787          81812           32023

unique        24871  ...             2             13              31

top         Private  ...            No  Reciprocating             091

freq            240  ...         80312          69530           18221


       Schedule Purpose.of.flight Air.carrier Weather.Condition  \
count     12582             82697       16648             84397
unique        3                26       13590                 4
top        NSCH          Personal       Pilot               VMC
freq       4474             49448         258             77303

       Broad.phase.of.flight   Report.Status Publication.Date
count                  61724           82508            75118
unique                    12           17075             2924
top                  Landing  Probable Cause       25-09-2020
freq                   15428           61754            17019

[4 rows x 26 columns]
```

```
#This gives the total overview of our dataset
#From the look of the data, we have a couple of missing data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88889 entries, 0 to 88888
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Event.Id                88889 non-null  object
 1   Investigation.Type      88889 non-null  object
 2   Accident.Number         88889 non-null  object
 3   Event.Date              88889 non-null  object
 4   Location                88837 non-null  object
 5   Country                 88663 non-null  object
 6   Latitude                34382 non-null  object
 7   Longitude               34373 non-null  object
 8   Airport.Code            50249 non-null  object
 9   Airport.Name            52790 non-null  object
 10  Injury.Severity         87889 non-null  object
 11  Aircraft.damage         85695 non-null  object
 12  Aircraft.Category       32287 non-null  object
 13  Registration.Number     87572 non-null  object
```

```
 14  Make                    88826 non-null  object
 15  Model                   88797 non-null  object
 16  Amateur.Built           88787 non-null  object
 17  Number.of.Engines       82805 non-null  float64
 18  Engine.Type             81812 non-null  object
 19  FAR.Description         32023 non-null  object
 20  Schedule                12582 non-null  object
 21  Purpose.of.flight       82697 non-null  object
 22  Air.carrier             16648 non-null  object
 23  Total.Fatal.Injuries    77488 non-null  float64
 24  Total.Serious.Injuries  76379 non-null  float64
 25  Total.Minor.Injuries    76956 non-null  float64
 26  Total.Uninjured         82977 non-null  float64
 27  Weather.Condition       84397 non-null  object
 28  Broad.phase.of.flight   61724 non-null  object
 29  Report.Status           82508 non-null  object
 30  Publication.Date        75118 non-null  object
dtypes: float64(5), object(26)
memory usage: 21.0+ MB
```

```
#Now lets find the missing values
#Well,we have a number of collumns that have missing values
df.isna().sum()
```

```
Event.Id                      0
Investigation.Type            0
Accident.Number               0
Event.Date                    0
Location                     52
Country                     226
Latitude                  54507
Longitude                 54516
Airport.Code              38640
Airport.Name              36099
Injury.Severity            1000
Aircraft.damage            3194
Aircraft.Category         56602
Registration.Number        1317
Make                         63
Model                        92
Amateur.Built               102
Number.of.Engines          6084
Engine.Type                7077
FAR.Description           56866
Schedule                  76307
Purpose.of.flight          6192
Air.carrier               72241
Total.Fatal.Injuries      11401
Total.Serious.Injuries    12510
Total.Minor.Injuries      11933
```

```
Total.Uninjured            5912
Weather.Condition          4492
Broad.phase.of.flight     27165
Report.Status              6381
Publication.Date          13771
dtype: int64
```

```python
#Lets now check the percentage of missing values in the columns so as it can give us insights on what columns we would drop
df.isna().sum()/len(df)*100
```

```
Event.Id                   0.000000
Investigation.Type         0.000000
Accident.Number            0.000000
Event.Date                 0.000000
Location                   0.058500
Country                    0.254250
Latitude                  61.320298
Longitude                 61.330423
Airport.Code              43.469946
Airport.Name              40.611324
Injury.Severity            1.124999
Aircraft.damage            3.593246
Aircraft.Category         63.677170
Registration.Number        1.481623
Make                       0.070875
Model                      0.103500
Amateur.Built              0.114750
Number.of.Engines          6.844491
Engine.Type                7.961615
FAR.Description           63.974170
Schedule                  85.845268
Purpose.of.flight          6.965991
Air.carrier               81.271023
Total.Fatal.Injuries      12.826109
Total.Serious.Injuries    14.073732
Total.Minor.Injuries      13.424608
Total.Uninjured            6.650992
Weather.Condition          5.053494
Broad.phase.of.flight     30.560587
Report.Status              7.178616
Publication.Date          15.492356
dtype: float64
```

Great, We now know what columns have massive numbers of missing data and dropping them will be neccessary. Not droping them would affect what I really want to achieve.

Cleaning the dataset

droping the collumns that wont be needed. All the columns that I would be droping I will be storing them in the columns_dropped variable.

```
#placing unneccessary columns in a variable
columns_dropped = ['Accident.Number','Location', 'Country','Latitude',
'Longitude', 'Airport.Code', 'Airport.Name', 'Registration.Number',
'FAR.Description',
'Schedule','Air.carrier', 'Publication.Date']

#droping the columns from the original dataframe to refined one
cleaned_df = df.drop(columns=columns_dropped)

#first five rows of the dataset
cleaned_df.head()
```

```
         Event.Id Investigation.Type  Event.Date Injury.Severity  \
0  20001218X45444           Accident  1948-10-24        Fatal(2)
1  20001218X45447           Accident  1962-07-19        Fatal(4)
2  20061025X01555           Accident  1974-08-30        Fatal(3)
3  20001218X45448           Accident  1977-06-19        Fatal(2)
4  20041105X01764           Accident  1979-08-02        Fatal(1)

  Aircraft.damage Aircraft.Category      Make     Model Amateur.Built
\
0       Destroyed               NaN   Stinson     108-3            No

1       Destroyed               NaN     Piper  PA24-180            No

2       Destroyed               NaN    Cessna      172M            No

3       Destroyed               NaN  Rockwell       112            No

4       Destroyed               NaN    Cessna       501            No


   Number.of.Engines    Engine.Type Purpose.of.flight
Total.Fatal.Injuries  \
0                1.0  Reciprocating          Personal
2.0
1                1.0  Reciprocating          Personal
4.0
2                1.0  Reciprocating          Personal
3.0
3                1.0  Reciprocating          Personal
2.0
4                NaN            NaN          Personal
1.0
```

```
   Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured  \
0                     0.0                   0.0              0.0
1                     0.0                   0.0              0.0
2                     NaN                   NaN              NaN
3                     0.0                   0.0              0.0
4                     2.0                   NaN              0.0

  Weather.Condition Broad.phase.of.flight   Report.Status
0               UNK                Cruise  Probable Cause
1               UNK               Unknown  Probable Cause
2               IMC                Cruise  Probable Cause
3               IMC                Cruise  Probable Cause
4               VMC              Approach  Probable Cause
```

```
#finding duplicates
cleaned_df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
         ...
88884    False
88885    False
88886    False
88887    False
88888    False
Length: 88889, dtype: bool
```

```
#Checking out the duplicates and printing them out
duplicates = cleaned_df[df.duplicated()]

print(duplicates)
```

```
Empty DataFrame
Columns: [Event.Id, Investigation.Type, Event.Date, Injury.Severity,
Aircraft.damage, Aircraft.Category, Make, Model, Amateur.Built,
Number.of.Engines, Engine.Type, Purpose.of.flight,
Total.Fatal.Injuries, Total.Serious.Injuries, Total.Minor.Injuries,
Total.Uninjured, Weather.Condition, Broad.phase.of.flight,
Report.Status]
Index: []
```

```
#Choosing a primary key which helps in removing duplicates
cleaned_df = cleaned_df.drop_duplicates(subset="Event.Id")

#Now lets do a quick check at our dataframe to see if there are
duplicates
duplicate_count = cleaned_df.duplicated().sum()
```

```
duplicate_count
print(f"Number of duplicate rows: {duplicate_count}")
```

```
Number of duplicate rows: 0
```

Great, We dropped the duplicates.

Data Analysis

```
#This line basically ignores the warnings
import warnings
warnings.filterwarnings('ignore')
#This defines a function called aircraft_category and identifies the
Nan values and replaces them with Unknown
def aircraft_category(value):
    if pd.isna(value) or value.strip() == "":
        return "Unknown"
    return value.strip().title()

# Apply it to the column
cleaned_df["Aircraft.Category"] =
cleaned_df["Aircraft.Category"].apply(aircraft_category)

#to check on the first 5 rows
cleaned_df.head()
```

```
         Event.Id Investigation.Type  Event.Date Injury.Severity  \
0  20001218X45444           Accident  1948-10-24        Fatal(2)
1  20001218X45447           Accident  1962-07-19        Fatal(4)
2  20061025X01555           Accident  1974-08-30        Fatal(3)
3  20001218X45448           Accident  1977-06-19        Fatal(2)
4  20041105X01764           Accident  1979-08-02        Fatal(1)

  Aircraft.damage Aircraft.Category      Make     Model Amateur.Built
\
0       Destroyed           Unknown   Stinson     108-3            No

1       Destroyed           Unknown     Piper  PA24-180            No

2       Destroyed           Unknown    Cessna      172M            No

3       Destroyed           Unknown  Rockwell       112            No

4       Destroyed           Unknown    Cessna       501            No


   Number.of.Engines    Engine.Type Purpose.of.flight
Total.Fatal.Injuries  \
0                1.0  Reciprocating          Personal
```

```
2.0
1                  1.0  Reciprocating          Personal
4.0
2                  1.0  Reciprocating          Personal
3.0
3                  1.0  Reciprocating          Personal
2.0
4                  NaN            NaN          Personal
1.0

   Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured  \
0                     0.0                   0.0              0.0
1                     0.0                   0.0              0.0
2                     NaN                   NaN              NaN
3                     0.0                   0.0              0.0
4                     2.0                   NaN              0.0

  Weather.Condition Broad.phase.of.flight   Report.Status
0               UNK                Cruise  Probable Cause
1               UNK               Unknown  Probable Cause
2               IMC                Cruise  Probable Cause
3               IMC                Cruise  Probable Cause
4               VMC              Approach  Probable Cause
```

Now, lets check for the other existing nan values

```
#Checks for the number of nan values in each and every column in my cleaned_df
cleaned_df.isna().sum()
```

```
Event.Id                      0
Investigation.Type            0
Event.Date                    0
Injury.Severity             990
Aircraft.damage            3103
Aircraft.Category             0
Make                         63
Model                        92
Amateur.Built               100
Number.of.Engines          6027
Engine.Type                7024
Purpose.of.flight          6122
Total.Fatal.Injuries      11267
Total.Serious.Injuries    12322
Total.Minor.Injuries      11760
Total.Uninjured            5863
Weather.Condition          4473
Broad.phase.of.flight     27114
```

```
Report.Status                  6361
dtype: int64

#This if and ifelse loop checks out our data set and replaces the nan
values with median for numerical values and mode for categorical
values
for column in cleaned_df.columns:
    if cleaned_df[column].dtype == 'object':
        #Categorical value filled with the most frequent value(mode)
        cleaned_df[column] =
cleaned_df[column].fillna(cleaned_df[column].mode()[0])
    else:
        #Numerical value filled with median
        cleaned_df[column] =
cleaned_df[column].fillna(cleaned_df[column].median())

cleaned_df.head()

         Event.Id Investigation.Type  Event.Date Injury.Severity  \
0  20001218X45444           Accident  1948-10-24        Fatal(2)
1  20001218X45447           Accident  1962-07-19        Fatal(4)
2  20061025X01555           Accident  1974-08-30        Fatal(3)
3  20001218X45448           Accident  1977-06-19        Fatal(2)
4  20041105X01764           Accident  1979-08-02        Fatal(1)

  Aircraft.damage Aircraft.Category      Make     Model Amateur.Built
\
0       Destroyed           Unknown   Stinson     108-3            No

1       Destroyed           Unknown     Piper  PA24-180            No

2       Destroyed           Unknown    Cessna      172M            No

3       Destroyed           Unknown  Rockwell       112            No

4       Destroyed           Unknown    Cessna       501            No


   Number.of.Engines    Engine.Type Purpose.of.flight
Total.Fatal.Injuries  \
0                1.0  Reciprocating          Personal
2.0
1                1.0  Reciprocating          Personal
4.0
2                1.0  Reciprocating          Personal
3.0
3                1.0  Reciprocating          Personal
2.0
4                1.0  Reciprocating          Personal
1.0
```

```
   Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured  \
0                     0.0                   0.0              0.0
1                     0.0                   0.0              0.0
2                     0.0                   0.0              1.0
3                     0.0                   0.0              0.0
4                     2.0                   0.0              0.0

  Weather.Condition Broad.phase.of.flight   Report.Status
0               UNK                Cruise  Probable Cause
1               UNK               Unknown  Probable Cause
2               IMC                Cruise  Probable Cause
3               IMC                Cruise  Probable Cause
4               VMC              Approach  Probable Cause
```

```
#Rechecking if there are any nan values
cleaned_df.isna().sum()
```

```
Event.Id                  0
Investigation.Type        0
Event.Date                0
Injury.Severity           0
Aircraft.damage           0
Aircraft.Category         0
Make                      0
Model                     0
Amateur.Built             0
Number.of.Engines         0
Engine.Type               0
Purpose.of.flight         0
Total.Fatal.Injuries      0
Total.Serious.Injuries    0
Total.Minor.Injuries      0
Total.Uninjured           0
Weather.Condition         0
Broad.phase.of.flight     0
Report.Status             0
dtype: int64
```

Great! the dataset does not nan values

```
import warnings
warnings.filterwarnings('ignore')
#The is a little bit of differences in values in my dataset. look at the Make column.Cessna tends to be the same as CESSNA.
#To fix this, I am making all the values in this column in capital letters
cleaned_df['Make'] = cleaned_df['Make'].str.upper()

cleaned_df.head()
```

```
          Event.Id Investigation.Type  Event.Date Injury.Severity  \
0  20001218X45444           Accident  1948-10-24        Fatal(2)
1  20001218X45447           Accident  1962-07-19        Fatal(4)
2  20061025X01555           Accident  1974-08-30        Fatal(3)
3  20001218X45448           Accident  1977-06-19        Fatal(2)
4  20041105X01764           Accident  1979-08-02        Fatal(1)

  Aircraft.damage Aircraft.Category      Make     Model Amateur.Built
\
0       Destroyed           Unknown   STINSON     108-3            No

1       Destroyed           Unknown     PIPER  PA24-180            No

2       Destroyed           Unknown    CESSNA      172M            No

3       Destroyed           Unknown  ROCKWELL       112            No

4       Destroyed           Unknown    CESSNA       501            No


   Number.of.Engines    Engine.Type Purpose.of.flight
Total.Fatal.Injuries  \
0                1.0  Reciprocating          Personal
2.0
1                1.0  Reciprocating          Personal
4.0
2                1.0  Reciprocating          Personal
3.0
3                1.0  Reciprocating          Personal
2.0
4                1.0  Reciprocating          Personal
1.0

   Total.Serious.Injuries  Total.Minor.Injuries  Total.Uninjured  \
0                     0.0                   0.0              0.0
1                     0.0                   0.0              0.0
2                     0.0                   0.0              1.0
3                     0.0                   0.0              0.0
4                     2.0                   0.0              0.0

  Weather.Condition Broad.phase.of.flight   Report.Status
0               UNK                Cruise  Probable Cause
1               UNK               Unknown  Probable Cause
2               IMC                Cruise  Probable Cause
3               IMC                Cruise  Probable Cause
4               VMC              Approach  Probable Cause
```

```
cleaned_df.shape
```

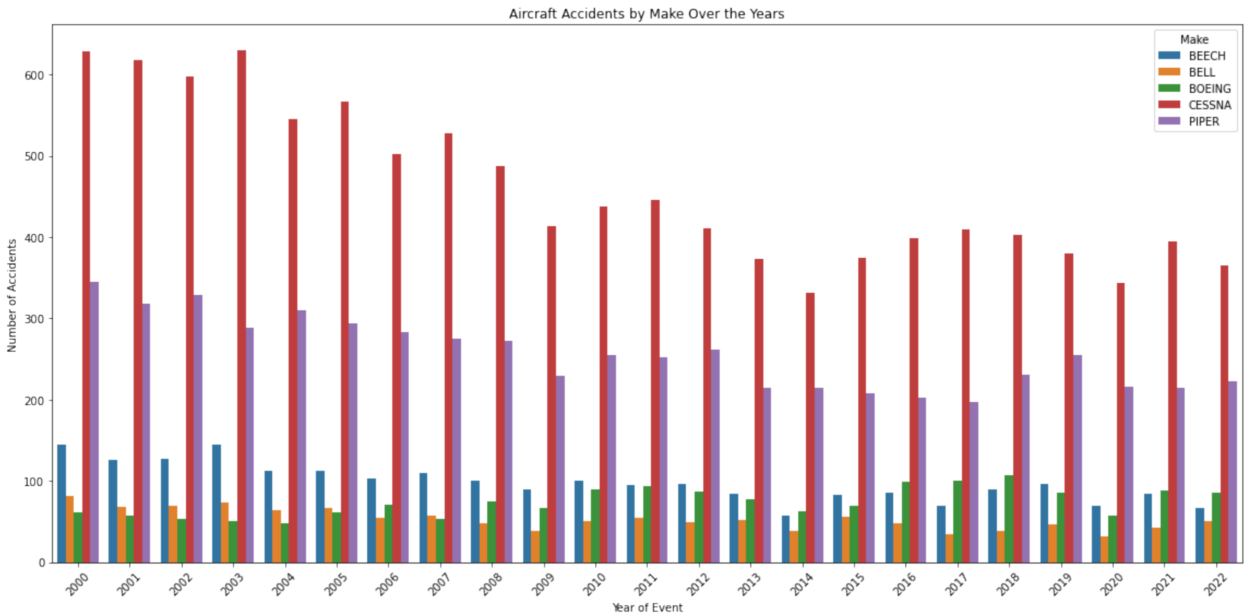
```
(87951, 19)
```

Great, the dataset is now clean.

Visualizations

Great, now that we have a much clearer dataset, let us now visualise it to make a more informed decision. First, let us plot a bargraph to see the number of accidents of all the aircrafts in our dataset over the years to see which aircraft has the most number of accidents.

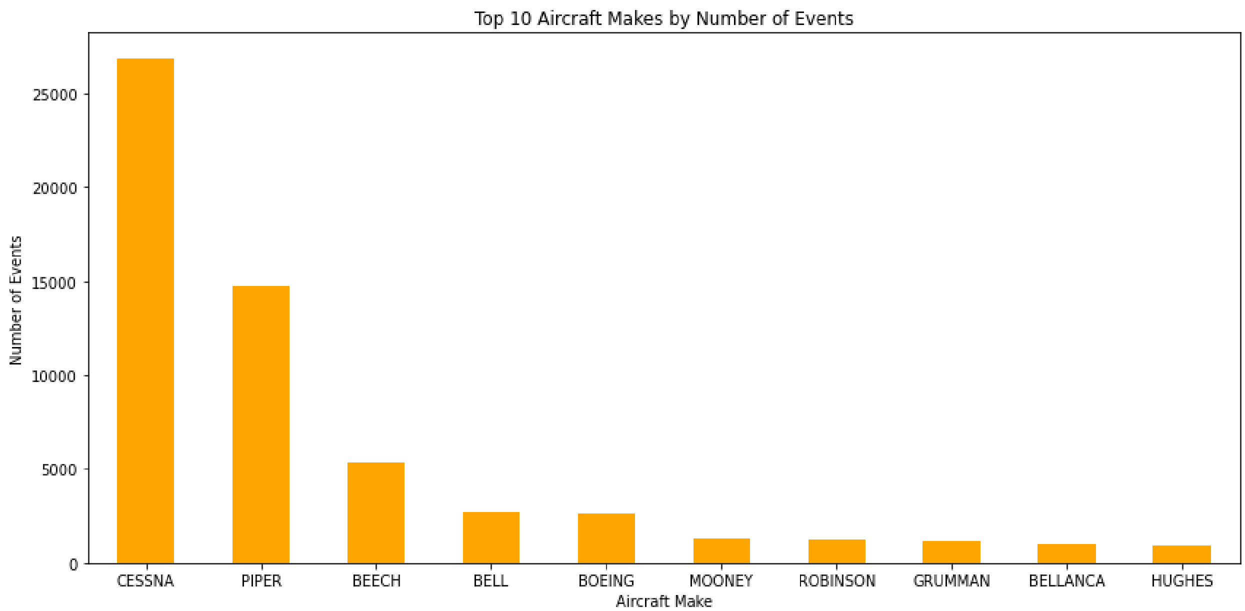
```
#creates a new dataframe called Event.Date and uses pandas to read it
as a date
cleaned_df['Event.Date'] = pd.to_datetime(cleaned_df['Event.Date'],
errors='coerce')

# Create Event.Year column
cleaned_df['Event.Year'] = cleaned_df['Event.Date'].dt.year
#Creates a new column that only contains the year part.This is why we
are using dt.year in my previous code.
grouped = cleaned_df.groupby(['Event.Year',
'Make']).size().reset_index(name='Accident Count')

# Optional: Filter recent years and top makes
grouped = grouped[grouped['Event.Year'] >= 2000]
top_makes = cleaned_df['Make'].value_counts().head(5).index
grouped = grouped[grouped['Make'].isin(top_makes)]

# Plot
plt.figure(figsize=(16,8))
ax = sns.barplot(data=grouped, x='Event.Year', y='Accident Count',
hue='Make', dodge=True, palette='tab10')

#Customize
ax.set_title('Aircraft Accidents by Make Over the Years')
ax.set_xlabel('Year of Event')
ax.set_ylabel('Number of Accidents')
ax.legend(title='Make', loc='upper right')
plt.xticks(rotation=45)
plt.tight_layout()

#saves image to my device
plt.savefig('Aircraft Accidents by Make Over the Years.png', dpi=300)
plt.show()
```



oh, Perfect we now know basing on this bargraph that Cessna has been having more accidents over the years.

Now,lets have a look at each make and the number of events based on the event ID.

```
# Group by Make and count
event_counts = cleaned_df.groupby('Make')
['Event.Id'].nunique().sort_values(ascending=False)

# Only take top 10
top_10_makes = event_counts.head(10)

# Plot
plt.figure(figsize=(12,6))
top_10_makes.plot(kind='bar', color='orange')

plt.title('Top 10 Aircraft Makes by Number of Events')
plt.xlabel('Aircraft Make')
plt.ylabel('Number of Events')
plt.xticks(rotation=360)

#saves image to my device
plt.savefig('Top 10 Aircraft Makes by Number of Events', dpi=300)
plt.tight_layout()
```

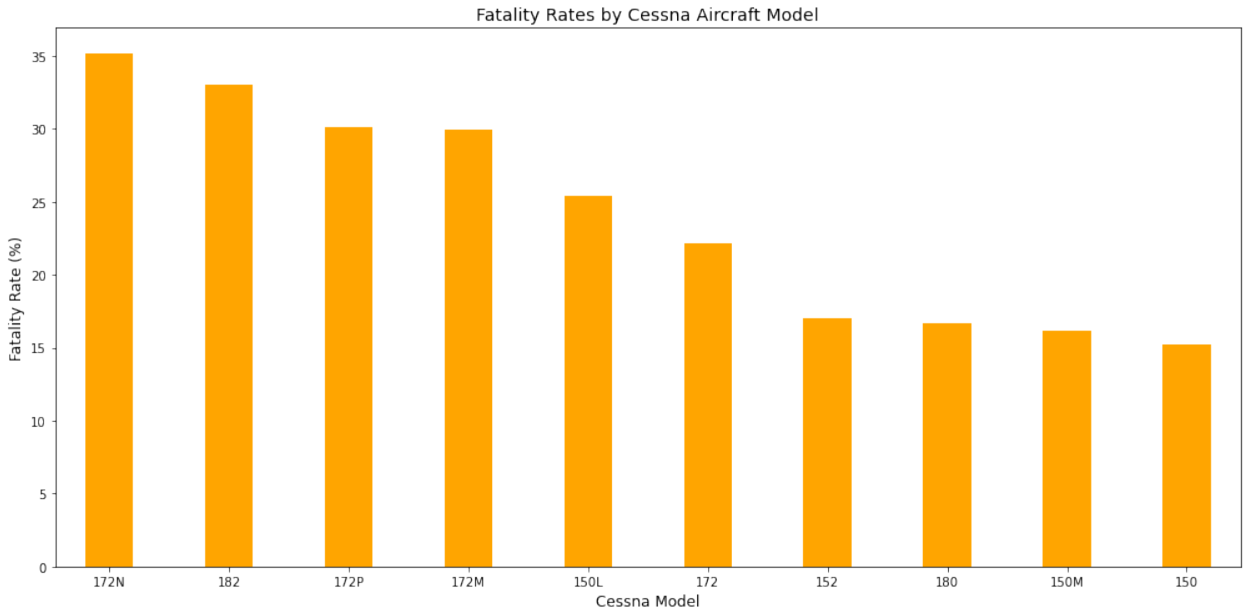


Again,CESSNA seems to have a huge number of events compaired to the rest of the aircrafts.

Lets take a look at CESSNA

```
# Filter data for CESSNA models
cessna_data = cleaned_df[cleaned_df['Make'] == 'CESSNA']

# Calculate the number of accidents for each Cessna model
accidents_by_model = cessna_data['Model'].value_counts().head(10)

# Calculate the total number of fatal injuries for each Cessna model
fatalities_by_model = cessna_data.groupby('Model')
['Total.Fatal.Injuries'].sum()

# Calculate the fatality rate (total fatal injuries / number of
accidents). multiplying it by 100 turns it to a percentage.
fatality_rate = (fatalities_by_model /
accidents_by_model).dropna().sort_values(ascending=False) * 100

# Plot the fatality rates for Cessna models
plt.figure(figsize=(14, 7))
plt.title('Fatality Rates by Cessna Aircraft Model', fontsize=14)
fatality_rate.plot(kind='bar', color='orange', width=0.4)
plt.xlabel('Cessna Model', fontsize=12)
plt.ylabel('Fatality Rate (%)', fontsize=12)
plt.xticks(rotation=360)

# Adjust layout to make room for labels
plt.tight_layout()

#saves image to my device
```

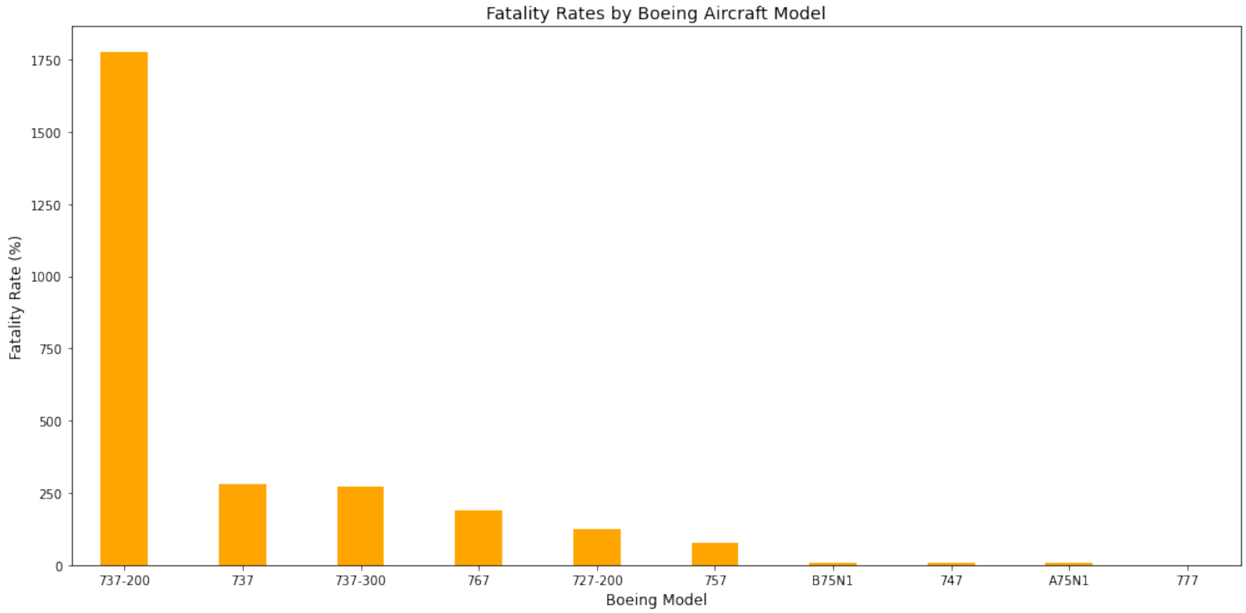
```
plt.savefig('Fatality Rates by Cessna Aircraft Model', dpi=300)
plt.show()
```



The most predominant model is the cessna 172N thereby making it the most used. Lets get a dive into BOEING.

```
# Filter data for Boeing models
boeing_data = cleaned_df[cleaned_df['Make'] == 'BOEING']

# Calculate the number of accidents for each Cessna model
accidents_by_model = boeing_data['Model'].value_counts().head(10)

# Calculate the total number of fatal injuries for each Cessna model
fatalities_by_model = boeing_data.groupby('Model')
['Total.Fatal.Injuries'].sum()

# Calculate the fatality rate (total fatal injuries / number of
accidents). multiplying it by 100 turns it to a percentage.
fatality_rate = (fatalities_by_model /
accidents_by_model).dropna().sort_values(ascending=False) * 100

# Plot the fatality rates for Cessna models
plt.figure(figsize=(14, 7))
plt.title('Fatality Rates by Boeing Aircraft Model', fontsize=14)
fatality_rate.plot(kind='bar', color='orange', width=0.4)
plt.xlabel('Boeing Model', fontsize=12)
plt.ylabel('Fatality Rate (%)', fontsize=12)
plt.xticks(rotation=360)

# Adjust layout to make room for labels
plt.tight_layout()
```

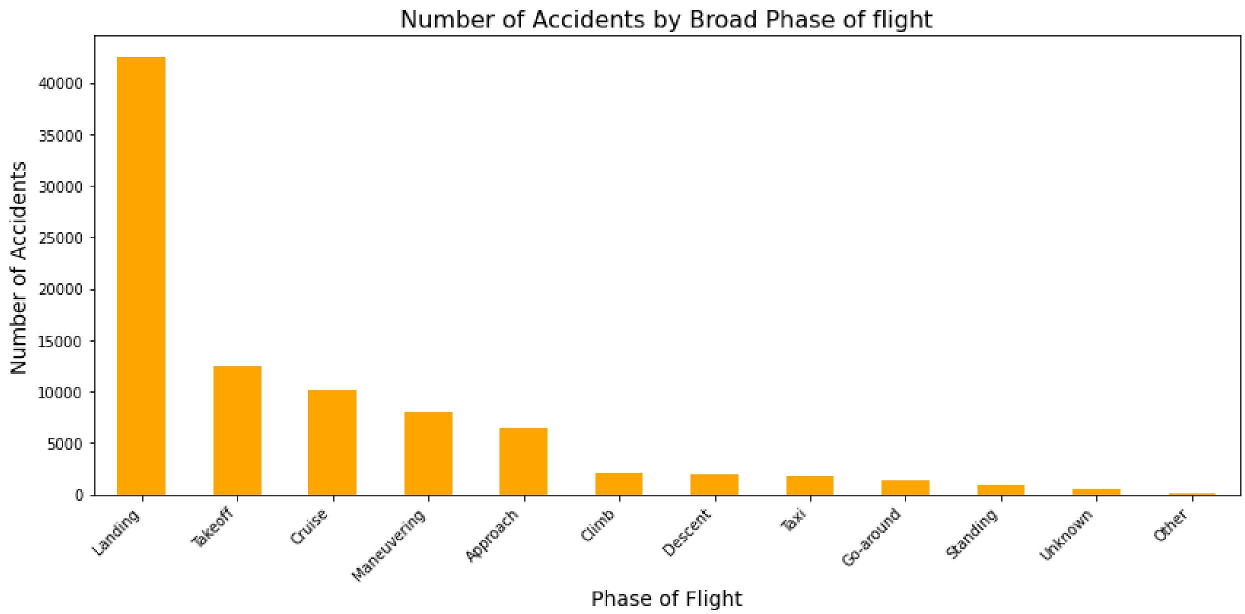
```
#saves image to my device
plt.savefig('Fatality Rates by boeing Aircraft Mode', dpi=300)
plt.show()
```



BOEING 737-200 seems to be the most predominant model based on the fatality rate thereby making it the most used model.

Now, lets check the number of accidents basing on the phase of flight.

```
# Groups by Broad Phase of Flight and count number of accidents
phase_counts = cleaned_df['Broad.phase.of.flight'].value_counts()

# Plot
fig, ax = plt.subplots(figsize=(12, 6))
phase_counts.plot(kind='bar', ax=ax, color='orange')

#Set titles and labels
ax.set_title('Number of Accidents by Broad Phase of flight',
fontsize=16)
ax.set_xlabel('Phase of Flight', fontsize=14)
ax.set_ylabel('Number of Accidents', fontsize=14)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

plt.tight_layout()

#saves image to my device
plt.savefig('Fatality Rates by boeing Aircraft Mode', dpi=300)
plt.show()
```



Most accidents occur during landing !!!

Recommendations.

1. Basing on this analysis, the Two most used models basing on the number of reported incidences are CESSNA and BOEING.
2. CESSNA'S most used model tends to be the 172 but it is not the safest model.Safe ones are the 180,150M,150
3. BOEING'S most used model tends to be 737-200 but it is not the safest model.Safe ones are the 747,A75N1 and 777
4. CESSNA is the most used airplane since 2000 to 2022
5. All the other planes are of low risk because they have less reported cases.

It would therefore be ideal if the company invests in this two airplane because they are the most used brands in the market thereby making them reliable. For short flights that only carry a smaller number of people, CESSNA would do an incredible job. BOEING would be ideal for long commercial flights that carry a good number of people. To avoid accidents, the company should always do regular check ups on both planes specifically on its landing ability and take off because this is when most accidents occur. Employing well experienced pilots would also avoid such incidences. All the other airplanes are also reliable if the company wouldnt want to consider market demand.

Convert my file to csv

```
#I am coverting my file to csv format.
df.to_csv("Cleaned_df.csv", index=False)
```